Given:

- plotData.m
- plotDecisionBoundary

Lab_03_LogisticReg_Lin.m

```matlab
%% Machine Learning
% Lab 3: Logistic Regression (Linear case)
% —— Admit students to university ——
%{
In this part of the exercise, you will build a logistic regression model to
predict whether a student gets admitted into a university.
Suppose that you are the administrator of a university department and
you want to determine each applicant's chance of admission based on their
results on two exams. You have historical data from previous applicants
that you can use as a training set for logistic regression. For each training
example, you have the applicant's scores on two exams and the admissions
decision.
%}

%% Initialization
clear ; close all; clc

%% Load Data
%  The first two columns contains the exam scores and the third column
%  contains the label.

%Data parsing
data = load('ex2data1.txt');
X = data(:, [1, 2]); % x1: exam 1 point; x2: exam 2 point
Y = data(:, 3);      % Passed or Failed

% Print out some data points
fprintf('First 10 examples from the dataset: \n');
fprintf(' x = [%.0f %.0f], y = %.0f \n', [X(1:10,:) Y(1:10,:)]');

%% Plotting
%  We start the exercise by first plotting the data to understand the
%  the problem we are working with.

fprintf(['Plotting data with + indicating (y = 1) examples and o ' ...
         'indicating (y = 0) examples.\n']);

plotData(X, Y);
% Put some labels
hold on;
% Labels and Legend
xlabel('Exam 1 score')
ylabel('Exam 2 score')
% Specified in plot order
legend('Admitted', 'Not admitted')
hold off;

%% Compute Cost and Gradient
% m: number of samples
% n: number of features
[m, n] = size(X);
% Additional Bias
```

```
X = [ones(m, 1) X];

% Initialize Weights
initial_w = zeros(n + 1, 1);

% Compute and display initial cost and gradient
[C, grad] = costFunction(initial_w, X, Y);

fprintf('Cost at initial weights (zeros): %f\n', C);
fprintf('Expected cost (approx): 0.693\n');
fprintf('Gradient at initial weights (zeros): \n');
fprintf(' %f \n', grad);
fprintf('Expected gradients (approx):\n -0.1000\n -12.0092\n -11.2628\n');

% Compute and display cost and gradient with non-zero theta
test_w = [-24; 0.2; 0.2];
[C, grad] = costFunction(test_w, X, Y);

fprintf('\nCost at test weights: %f\n', C);
fprintf('Expected cost (approx): 0.218\n');
fprintf('Gradient at test weights: \n');
fprintf(' %f \n', grad);
fprintf('Expected gradients (approx):\n 0.043\n 2.566\n 2.647\n');

%% === Optimizing using fminunc ===

%  In this exercise, you will use a built-in function (fminunc) to find the
%  optimal parameters weights.

%  Set options for fminunc
options = optimset('GradObj', 'on', 'MaxIter', 400);

%  Run fminunc to obtain the optimal theta
%  This function will return weights and the cost
[w, C] = fminunc(@(t)(costFunction(t, X, Y)), initial_w, options);

% Print weights to screen
fprintf('Cost at weights found by fminunc: %f\n', C);
fprintf('Expected cost (approx): 0.203\n');
fprintf('weights: \n');
fprintf(' %f \n', w);
fprintf('Expected weights (approx):\n');
fprintf(' -25.161\n 0.206\n 0.201\n');

%% Plot Boundary
plotDecisionBoundary(w, X, Y);

% Put some labels
hold on;
% Labels and Legend
xlabel('Exam 1 score')
ylabel('Exam 2 score')

% Specified in plot order
legend('Admitted', 'Not admitted')
hold off;
```

```
%% ============== Part 4: Predict and Accuracies ==============
%  After learning the parameters, you'll like to use it to predict the outcomes
%  on unseen data. In this part, you will use the logistic regression model
%  to predict the probability that a student with score 45 on exam 1 and
%  score 85 on exam 2 will be admitted.
%
%  Furthermore, you will compute the training and test set accuracies of
%  our model.

prob = sigmoid([1 45 85] * w);
fprintf(['For a student with scores 45 and 85, we predict an admission ' ...
         'probability of %f\n'], prob);
fprintf('Expected value: 0.775 +/- 0.002\n\n');

% Compute accuracy on our training set
p = predict(w, X);

fprintf('Train Accuracy: %f\n', mean(double(p == Y)) * 100);
fprintf('Expected accuracy (approx): 89.0\n');
fprintf('\n');
```

sigmoid.m

```
function g = sigmoid(z)
%SIGMOID Compute sigmoid function
g = zeros(size(z));
g = 1 ./ (1 + exp(-z));
end
```

costFunction.m

```
function [C, grad] = costFunction(w, X, Y)
%COSTFUNCTION Compute cost and gradient for logistic regression
% Initialize some useful values
m = length(Y); % number of training examples
C = 0;
grad = zeros(size(w));

C = (1/m) * sum((-Y).*(log(sigmoid(X*w))) - (1-Y).*(log(1-sigmoid(X*w))));

for i=1:size(w)
  grad(i) = (1/m) * sum((sigmoid(X*w)-Y).*X(:,i));
end
end
```

predict.m

```
function p = predict(w, X)
%PREDICT Predict whether the label is 0 or 1 using learned logistic
m = size(X, 1); % Number of training examples
p = zeros(m, 1);

temp = sigmoid(X*w);
for i=1:m
  if temp(i)>=0.5
    temp(i) = 1;
  else
    temp(i) = 0;
  end
end
p = temp;
end
```